IN THE SPECIFICATION:

Please amend the paragraph that appears on page 4, line 15 through page 5, line 6 as follows:

In modern times, in which distributed networks (of which the Internet is merely one example) are ubiquitous, it is common for one application program that uses the same data as a second application program to require the data to be in a different format. For instance, two different vendors may provide software that a customer may wish to integrate into a larger computing system in which the software from one of the vendors generates data that will then be passed to the software of the other vendor for further processing. However, these two vendors may use different formats for data. Accordingly, many transport mechanisms are platform and/or format independent in order to provide a mechanism by which data can be transported between two computing entities that may use different platforms and/or data formats. Accordingly, data generated by one application program in a first format can be embedded in one of these platform and format independent transport mechanisms for transport.

Please amend the paragraph that appears on page 5, lines 7 through 13 as follows:

The receiving application program typically will have a front-end software module, e.g., an API, ~~may be~~ designed to, not only intercept the transport mechanism and parse the data therefrom before it reaches the actual application program tools, but that also determines the format of the received data and, if it is in an external format (i.e., a format other than the internal format used by the

application program tools of the receiving entity), converts it to the internal format used by the application program tools.

Please amend the paragraph that appears on page 7, lines 16 through 18 as follows:

In this routine, the entire XMI document within which the requested object is contained is retrieved, converted, parsed and written to the cache, even though only a single object was requested.

Please amend the paragraph that appears on page 9, lines 10 through 15 as follows:

The method and apparatus further allow the registration of reflection adapter factories and specialized model classes which work with those adapters for the purpose of computing object attributes, which would normally have been parsed from an XMI document, and populating the Java package resource with them. One implementation of this mechanism is a set of Java Reflection adapters which compute the properties of JavaClass model objects from the JDK reflection ~~API's~~ APIs.

Please amend the paragraph that appears on page 11, line 13 through page 12, line 6 as follows:

The method and apparatus allow the registration of resource factories which can retrieve the necessary data from a data source and create resources on the fly through mechanisms other than the traditional parsing of a file. The data source can comprises almost any form, such as a database, a document in ~~an~~ a format other than the transport mechanism format or ~~querying~~ a live system). Accordingly,

resources are not stored in memory as documents to be parsed when a request for data is received. The invention further provides a reflection mechanism for populating a complex object model implementation with data in the internal format of the requesting entity based on metadata in an external format contained in the transport mechanism document. The object model is populated on the fly responsive to a request for the data. The reflection mechanism is lazy and only those classes of the model that comprise the requested data are populated. Accordingly, an entire document need not be retrieved, converted in format, and parsed for every request for data, as was the case in the prior art.

Please amend the paragraph that appears on page 16, lines 14 through 16 as follows:

Specifically, very little is actually done ~~is~~ in step 208. Particularly, the object, "String~~,~~" was created with only enough information about it so that we can go back and get the rest of the information about the class when needed.

Please amend the paragraph that appears on page 17, lines 1 through 8 as follows:

Figure 3 is a flow chart illustrating the processing responsive to a tool=s request for such information and particularly illustrates operation in response to an exemplary request for the methods associated with the class "String." The process starts in step 301. In step 302, responsive to the inputs by the user seeking the methods of the class "String", the tool issues ~~the~~ a request, namely, Astringclass.getMethods( )". Responsive to this request, flow proceeds to step 303, where the cache is queried to determine if the methods are set. If the methods are

set, no further processing is required and the methods list is returned to the tool (Step 311) and the process terminates (step 312).